PROJECT REPORT

ON

"Analysis of Sentiment in Social Media Posts"

Department of Computer
Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O.

GLA
UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC

Declaration

I hereby declare that the work which is being presented in the Project Report **"Analysis of Sentiment in Social Media Posts ",** in partial fulfillment of the requirements for Project is an authentic record of my own work carried under the supervision of Mr.Mohd.Amir Khan**, GLA University, Mathura**.

NAME OF THE CANDIDATE : JAYANT BHARDWAJ

UNIVERSITY ROLL NO: 201500315

NAME OF THE CANDIDATE : LAKSHYA SHARMA

UNIVERSITY ROLL NO: 201500368

NAME OF THE CANDIDATE : ARYAN SINGH

UNIVERSITY ROLL NO: 201500158

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the synopsis of the mini project undertaken during B. Tech III Year. This project is going to be an acknowledgement to the inspiration, drive and technical assistance will be contributed to it by many individuals. We owe special debt of gratitude to Mr.Mohd. Amir khan sir , Technical Trainer, for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal and for him constant support and guidance to our work. Him sincerity, thoroughness and perseverance has been a constant source of inspiration for us. We believe that she will shower us with all him extensively experienced ideas and insightful comments at different stages of the project & also taught us about the latest industry-oriented technologies. We also do not like miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and co-operation.

ABOUT THE PROJECT

The project "Analysis of Sentiment in Social Media Posts using Machine Learning and Natural Language Processing" aims to develop a system that can automatically analyze the sentiment of social media posts using machine learning and natural language processing techniques. The system can collect social media posts from various sources, preprocess the text data, extract relevant features, train a sentiment classification model, and provide real-time sentiment analysis results on a web platform.

Sentiment analysis of social media posts is important for several reasons. Firstly, social media has become a significant source of public opinion, and sentiment analysis can help understand how people feel about a particular topic, product, or event. Secondly, sentiment analysis can help businesses or organizations monitor their brand reputation, identify customer needs or complaints, and improve their marketing or customer service strategies. Thirdly, sentiment analysis can help researchers or policymakers analyze public sentiment on political or social issues and make informed decisions.

TECHNOLOGY USED

MACHINE LEARNING

Machine learning is a subset of artificial intelligence that enables computer systems to automatically learn and improve from experience without being explicitly programmed. In machine learning, algorithms are trained on large datasets to recognize patterns and make predictions or decisions based on new data.Machine learning has a wide range of applications in various fields, including image recognition, natural language processing, speech recognition, recommendation systems, fraud detection, and predictive analytics.

The process of machine learning typically involves the following steps:

Data Collection: Collecting and preparing a large amount of data for the machine learning model to learn from.

Data Preprocessing: Cleaning and transforming the data to make it suitable for the machine learning model.

Model Selection: Choosing the appropriate machine learning model that can perform the desired task.

Training: Feeding the prepared data into the machine learning model to train it on the patterns in the data.

Evaluation: Testing the performance of the machine learning model on a separate dataset to evaluate its accuracy.

Deployment: Implementing the machine learning model into a real-world system to perform the desired task.

PYTHON: Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

NATURAL LANGUAGE PROCESSING:

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiments.

REPORT

Our project "Sentiment analysis of social media post using machine learnig" is completed as we have done data preprocessing on a certain dataset and showed the result of the sentiments of people on social media posts. In this project we have learned the use of Python, machine learning and little NLP , we have proceded our data on Google colab. This project might be the greatest broke into learning human behaviour and media sentiments in the upcoming days as we will be able to analyse the perfectly.

Whole work is done under the grateful mentorship of Mr.Mohd.Amir khan

Thankyou

Group Members

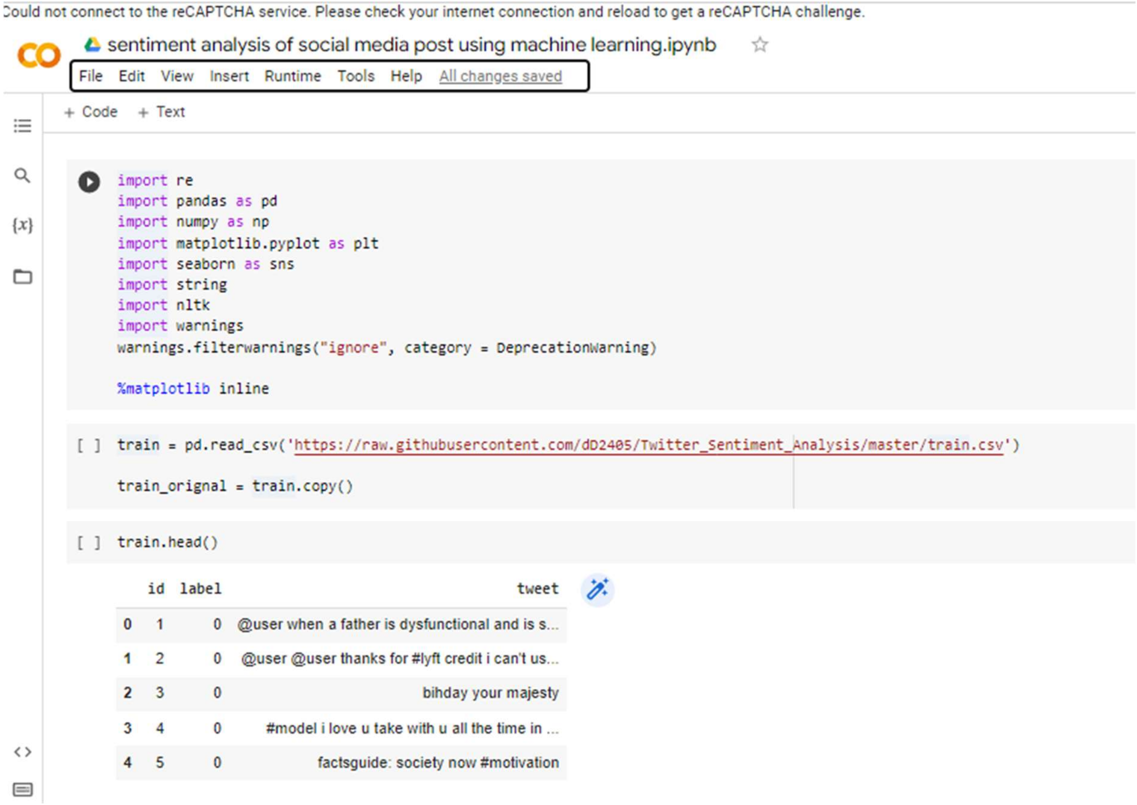
1. Jayant bhardwaj (201500315)
2. Lakshya sharma (201500368)
3. Aryan Singh(201500158)

Course: B.Tech (Computer Science and Engineering)

Year: 3rd

Semester: 6th

source code and snaps:



sentiment analysis of social media post using machine learning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

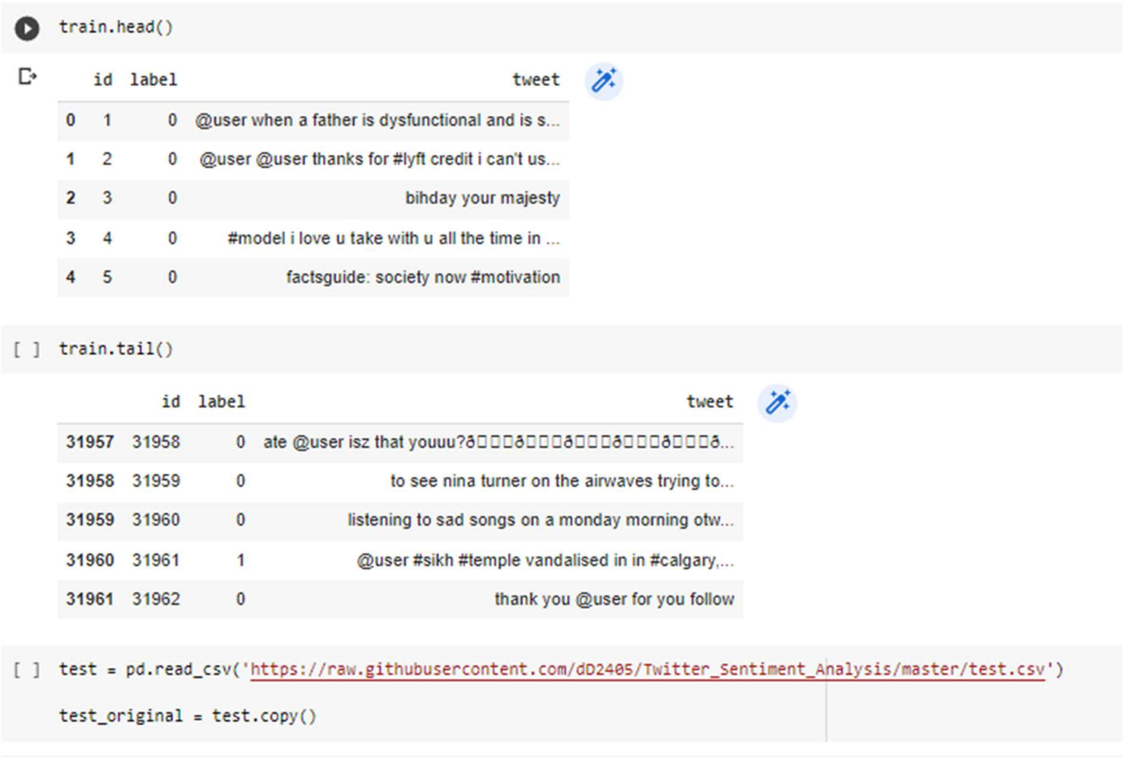
Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import string
import nltk
import warnings
warnings.filterwarnings("ignore", category = DeprecationWarning)

%matplotlib inline
```

```
[ ] train = pd.read_csv('https://raw.githubusercontent.com/dD2405/Twitter_Sentiment_Analysis/master/train.csv')

    train_orignal = train.copy()
```

```
[ ] train.head()
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation



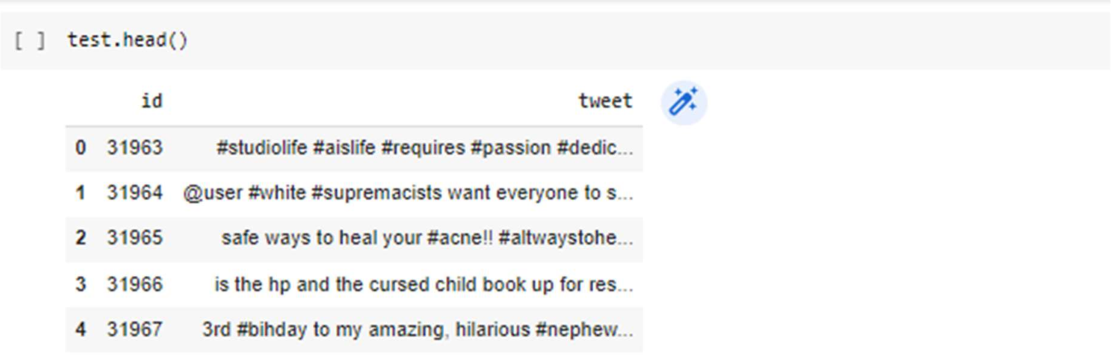
```
train.head()
```

```
   id  label                                              tweet
0   1      0  @user when a father is dysfunctional and is s...
1   2      0  @user @user thanks for #lyft credit i can't us...
2   3      0                                bihday your majesty
3   4      0  #model i love u take with u all the time in ...
4   5      0               factsguide: society now #motivation
```

```
[ ] train.tail()
```

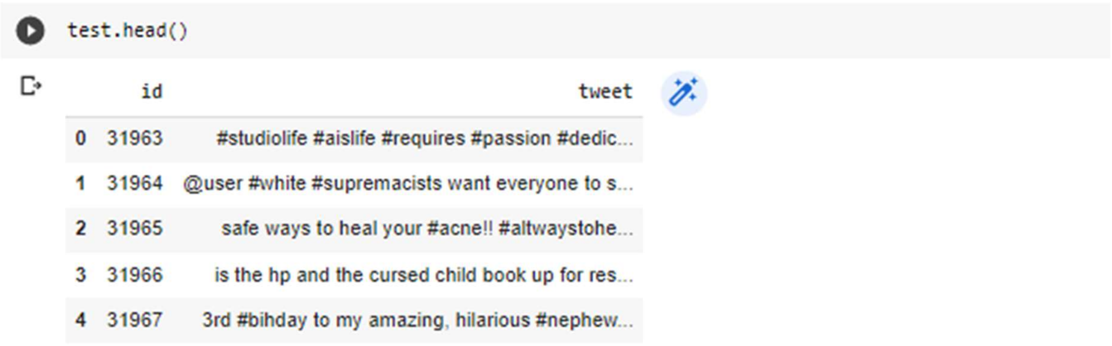
```
          id  label                                              tweet
31957  31958      0  ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð□□□ð...
31958  31959      0          to see nina turner on the airwaves trying to...
31959  31960      0      listening to sad songs on a monday morning otw...
31960  31961      1         @user #sikh #temple vandalised in in #calgary,...
31961  31962      0                           thank you @user for you follow
```

```
[ ] test = pd.read_csv('https://raw.githubusercontent.com/dD2405/Twitter_Sentiment_Analysis/master/test.csv')

    test_original = test.copy()
```



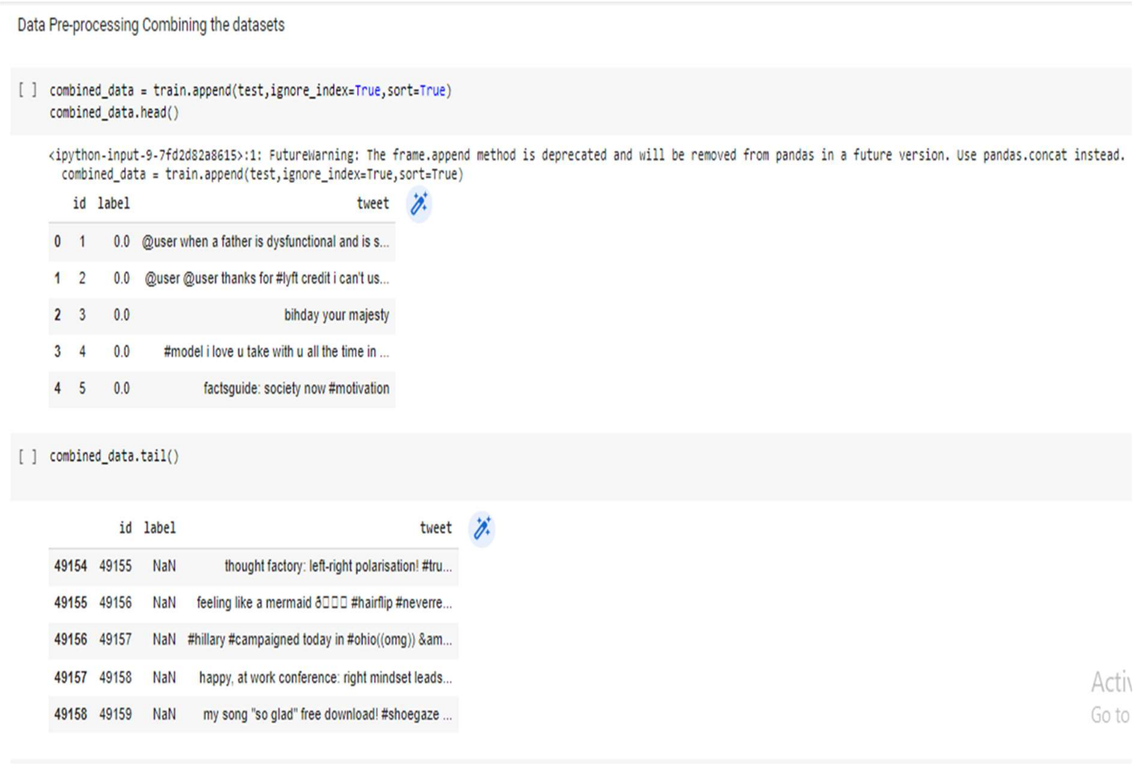
```
[ ] test.head()
```

```
      id                                             tweet
0  31963  #studiolife #aislife #requires #passion #dedic...
1  31964  @user #white #supremacists want everyone to s...
2  31965     safe ways to heal your #acne!! #altwaystohe...
3  31966     is the hp and the cursed child book up for res...
4  31967   3rd #bihday to my amazing, hilarious #nephew...
```



```
test.head()
```

```
      id                                             tweet
0  31963  #studiolife #aislife #requires #passion #dedic...
1  31964  @user #white #supremacists want everyone to s...
2  31965     safe ways to heal your #acne!! #altwaystohe...
3  31966     is the hp and the cursed child book up for res...
4  31967   3rd #bihday to my amazing, hilarious #nephew...
```



Data Pre-processing Combining the datasets

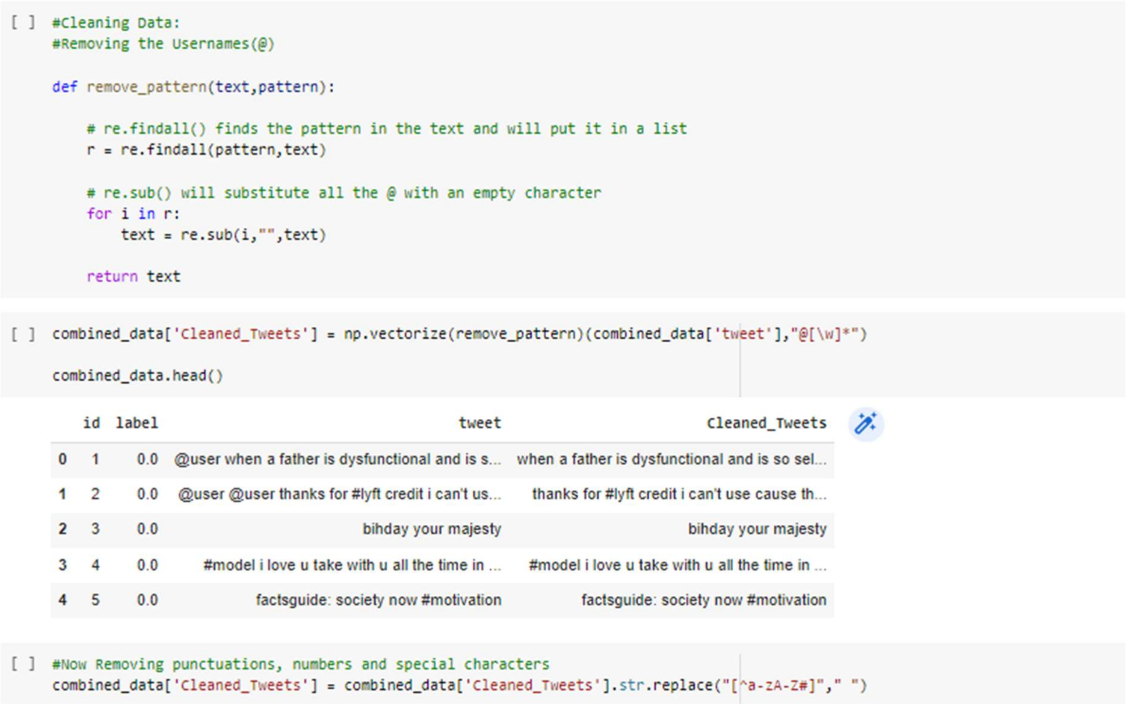
```
[ ] combined_data = train.append(test,ignore_index=True,sort=True)
    combined_data.head()
```

```
<ipython-input-9-7fd2d82a8615>:1: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
  combined_data = train.append(test,ignore_index=True,sort=True)
```

	id	label	tweet
0	1	0.0	@user when a father is dysfunctional and is s...
1	2	0.0	@user @user thanks for #lyft credit i can't us...
2	3	0.0	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation

```
[ ] combined_data.tail()
```

	id	label	tweet
49154	49155	NaN	thought factory: left-right polarisation! #tru...
49155	49156	NaN	feeling like a mermaid ð□□□ #hairflip #neverre...
49156	49157	NaN	#hillary #campaigned today in #ohio((omg)) &am...
49157	49158	NaN	happy, at work conference: right mindset leads...
49158	49159	NaN	my song "so glad" free download! #shoegaze ...

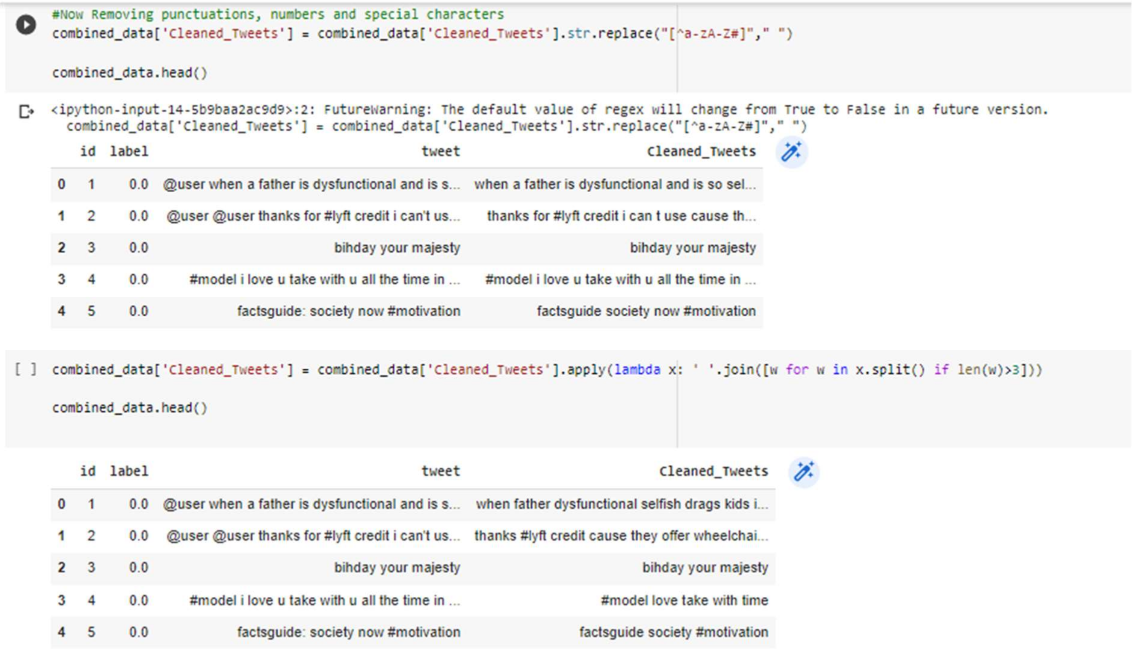


```
[ ] #Cleaning Data:
    #Removing the Usernames(@)

    def remove_pattern(text,pattern):

        # re.findall() finds the pattern in the text and will put it in a list
        r = re.findall(pattern,text)

        # re.sub() will substitute all the @ with an empty character
        for i in r:
            text = re.sub(i,"",text)

        return text
```

```
[ ] combined_data['Cleaned_Tweets'] = np.vectorize(remove_pattern)(combined_data['tweet'],"@[\w]*")

    combined_data.head()
```

	id	label	tweet	Cleaned_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can't use cause th...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation	factsguide: society now #motivation

```
[ ] #Now Removing punctuations, numbers and special characters
    combined_data['Cleaned_Tweets'] = combined_data['Cleaned_Tweets'].str.replace("[^a-zA-Z#]"," ")
```



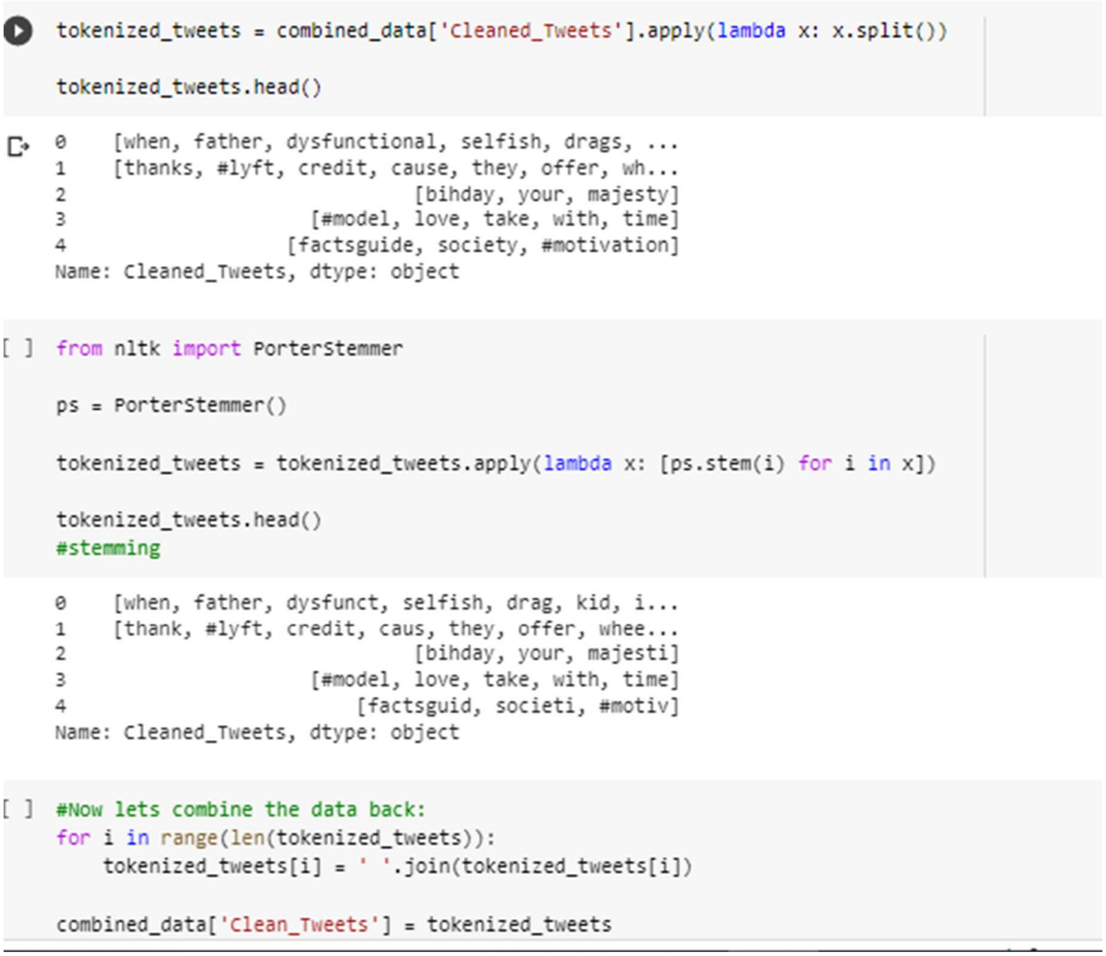
```
#Now Removing punctuations, numbers and special characters
combined_data['Cleaned_Tweets'] = combined_data['Cleaned_Tweets'].str.replace("[^a-zA-Z#]"," ")

combined_data.head()
```

```
<ipython-input-14-5b9baa2ac9d9>:2: FutureWarning: The default value of regex will change from True to False in a future version.
  combined_data['Cleaned_Tweets'] = combined_data['Cleaned_Tweets'].str.replace("[^a-zA-Z#]"," ")
```

	id	label	tweet	Cleaned_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can t use cause th...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0.0	factsguide: society now #motivation	factsguide society now #motivation

```
combined_data['Cleaned_Tweets'] = combined_data['Cleaned_Tweets'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))

combined_data.head()
```

	id	label	tweet	Cleaned_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit cause they offer wheelchai...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguide society #motivation



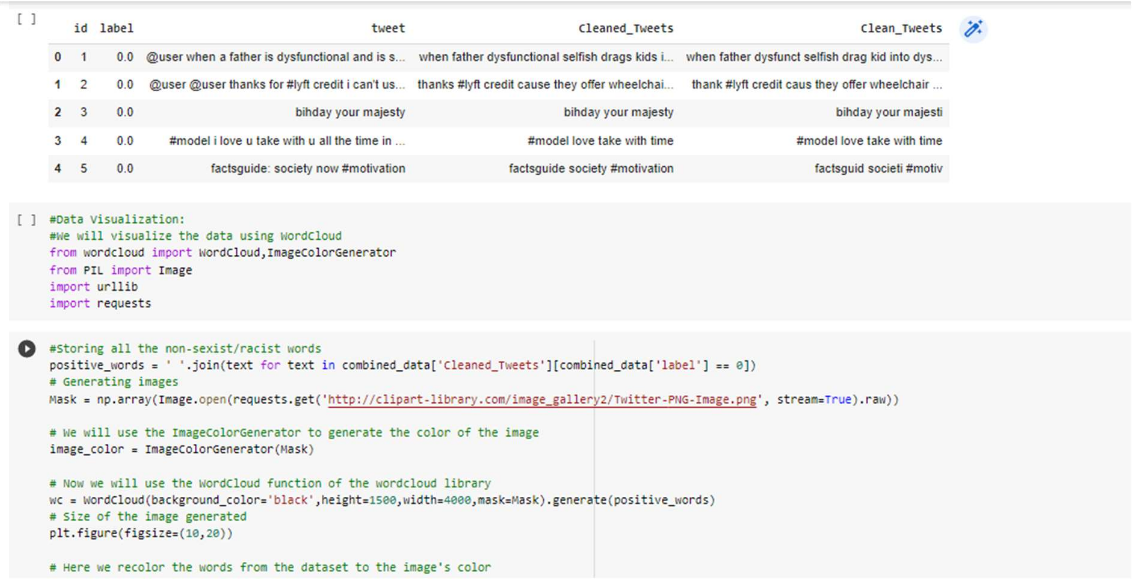
```
tokenized_tweets = combined_data['Cleaned_Tweets'].apply(lambda x: x.split())

tokenized_tweets.head()
```

```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, cause, they, offer, wh...
2                             [bihday, your, majesty]
3                      [#model, love, take, with, time]
4                    [factsguide, society, #motivation]
Name: Cleaned_Tweets, dtype: object
```

```
from nltk import PorterStemmer

ps = PorterStemmer()

tokenized_tweets = tokenized_tweets.apply(lambda x: [ps.stem(i) for i in x])

tokenized_tweets.head()
#stemming
```

```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, caus, they, offer, whee...
2                             [bihday, your, majesti]
3                      [#model, love, take, with, time]
4                          [factsguid, societi, #motiv]
Name: Cleaned_Tweets, dtype: object
```

```
#Now lets combine the data back:
for i in range(len(tokenized_tweets)):
    tokenized_tweets[i] = ' '.join(tokenized_tweets[i])

combined_data['Clean_Tweets'] = tokenized_tweets
```

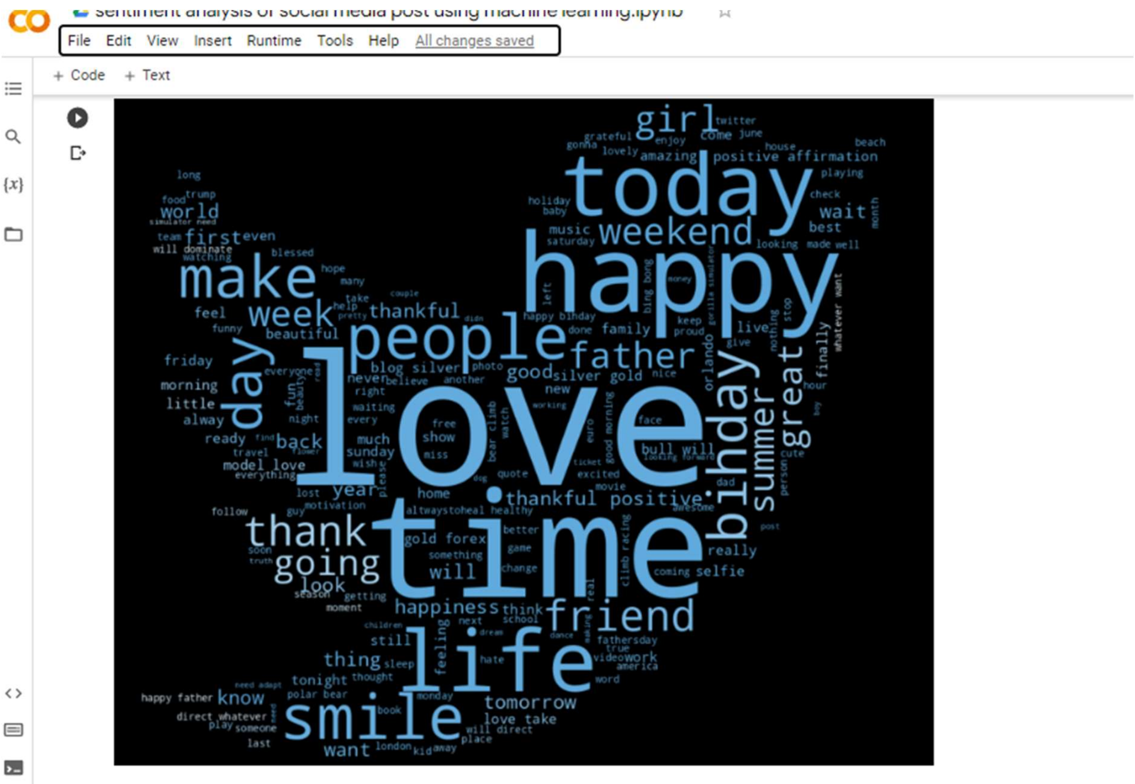


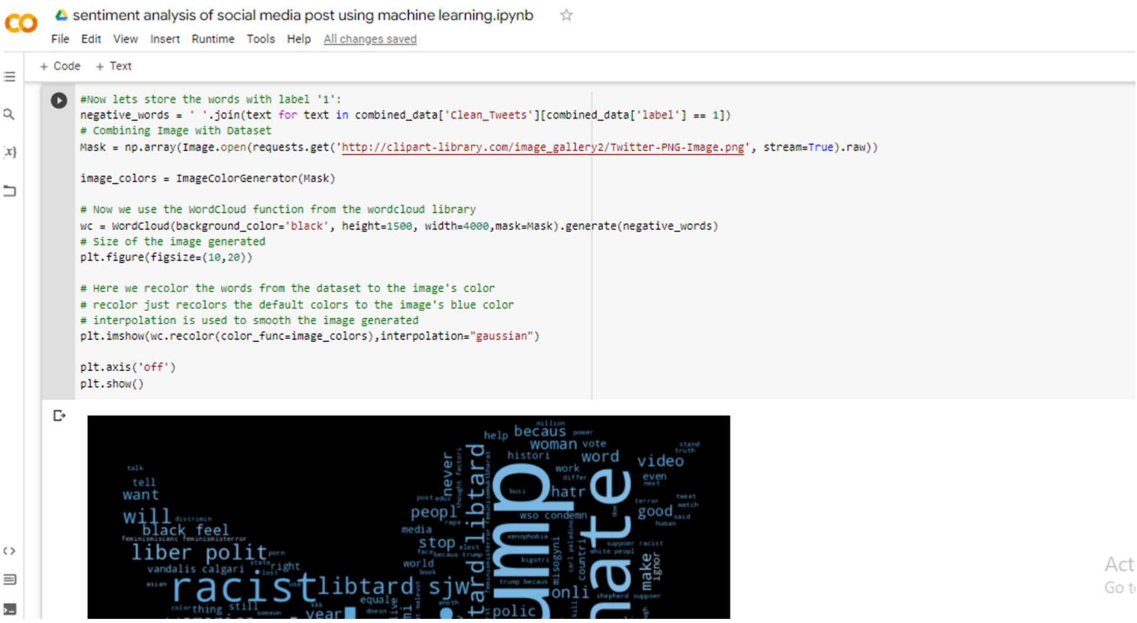
[ ]

	id	label	tweet	Cleaned_Tweets	Clean_Tweets
0	1	0.0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...	when father dysfunct selfish drag kid into dys...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit cause they offer wheelchai...	thank #lyft credit caus they offer wheelchair ...
2	3	0.0	bihday your majesty	bihday your majesty	bihday your majesti
3	4	0.0	#model i love u take with u all the time in ...	#model love take with time	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguide society #motivation	factsguid societi #motiv

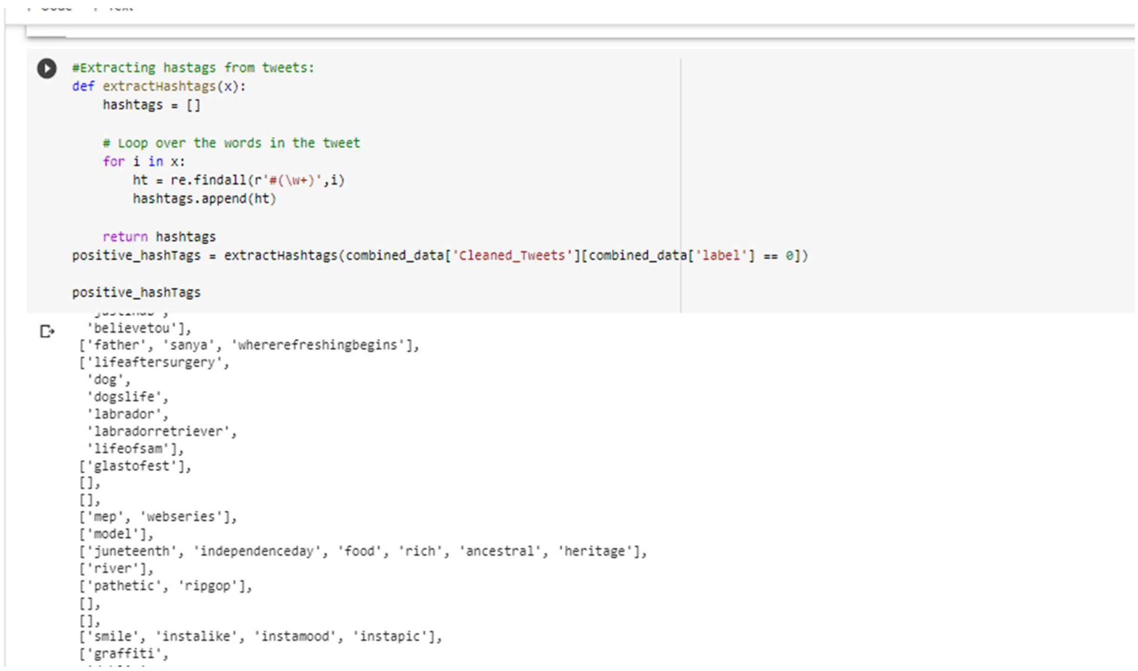
```
[ ] #Data Visualization:
    #We will visualize the data using WordCloud
    from wordcloud import WordCloud,ImageColorGenerator
    from PIL import Image
    import urllib
    import requests
```

```
#Storing all the non-sexist/racist words
positive_words = ' '.join(text for text in combined_data['Cleaned_Tweets'][combined_data['label'] == 0])
# Generating images
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitter-PNG-Image.png', stream=True).raw))

# We will use the ImageColorGenerator to generate the color of the image
image_color = ImageColorGenerator(Mask)

# Now we will use the WordCloud function of the wordcloud library
wc = WordCloud(background_color='black',height=1500,width=4000,mask=Mask).generate(positive_words)
# Size of the image generated
plt.figure(figsize=(10,20))

# Here we recolor the words from the dataset to the image's color
```



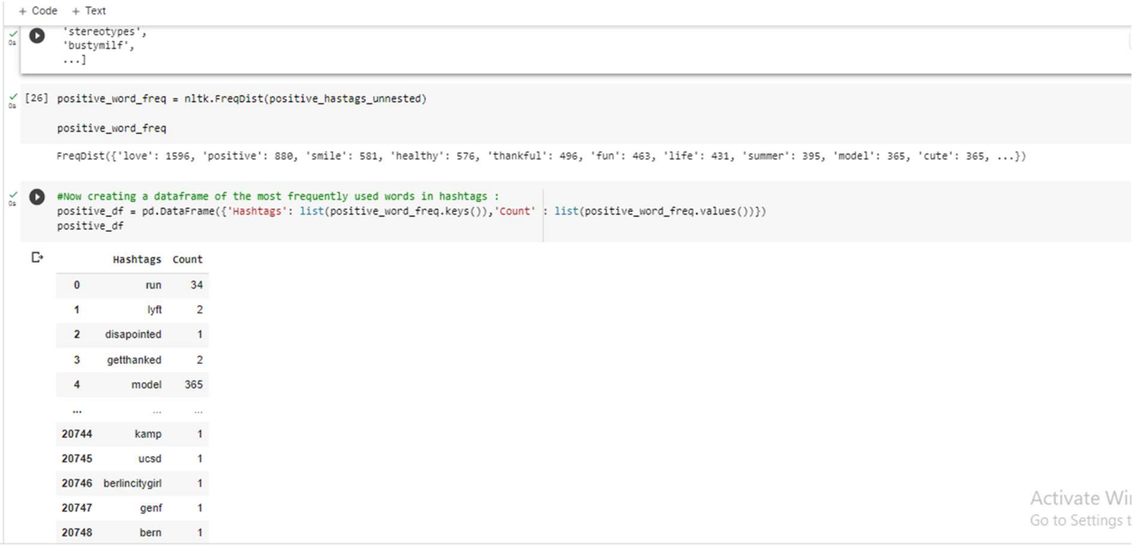


```
#Now lets store the words with label '1':
negative_words = ' '.join(text for text in combined_data['Clean_Tweets'][combined_data['label'] == 1])
# Combining Image with Dataset
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitter-PNG-Image.png', stream=True).raw))

image_colors = ImageColorGenerator(Mask)

# Now we use the WordCloud function from the wordcloud library
wc = WordCloud(background_color='black', height=1500, width=4000,mask=Mask).generate(negative_words)
# Size of the image generated
plt.figure(figsize=(10,20))

# Here we recolor the words from the dataset to the image's color
# recolor just recolors the default colors to the image's blue color
# interpolation is used to smooth the image generated
plt.imshow(wc.recolor(color_func=image_colors),interpolation="gaussian")

plt.axis('off')
plt.show()
```



```
#Extracting hastags from tweets:
def extractHashtags(x):
    hashtags = []

    # Loop over the words in the tweet
    for i in x:
        ht = re.findall(r'#(\w+)',i)
        hashtags.append(ht)

    return hashtags
positive_hashTags = extractHashtags(combined_data['Cleaned_Tweets'][combined_data['label'] == 0])

positive_hashTags
```

```
  'believetou'],
 ['father', 'sanya', 'whererefreshingbegins'],
 ['lifeaftersurgery',
  'dog',
  'dogslife',
  'labrador',
  'labradorretriever',
  'lifeofsam'],
 ['glastofest'],
 [],
 [],
 ['mep', 'webseries'],
 ['model'],
 ['juneteenth', 'independenceday', 'food', 'rich', 'ancestral', 'heritage'],
 ['river'],
 ['pathetic', 'ripgop'],
 [],
 [],
 ['smile', 'instalike', 'instamood', 'instapic'],
 ['graffiti',
```



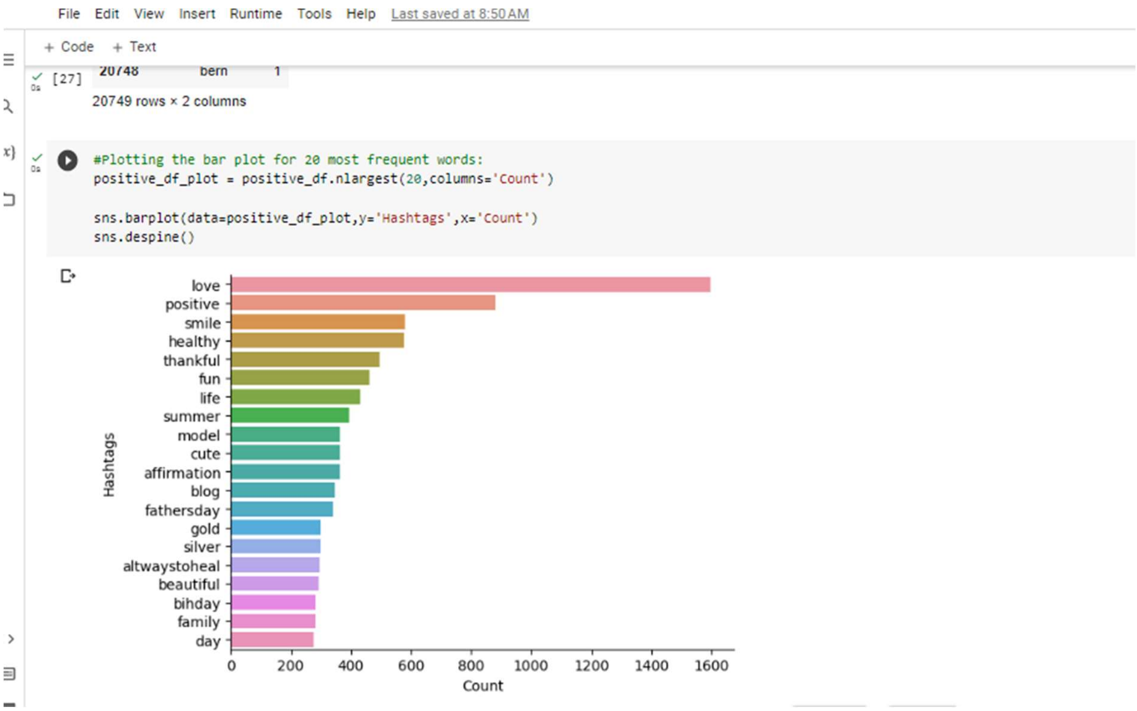
```
'stereotypes',
'bustymilf',
...]
```

```
[26] positive_word_freq = nltk.FreqDist(positive_hastags_unnested)

     positive_word_freq
```

```
FreqDist({'love': 1596, 'positive': 880, 'smile': 581, 'healthy': 576, 'thankful': 496, 'fun': 463, 'life': 431, 'summer': 395, 'model': 365, 'cute': 365, ...})
```

```
#Now creating a dataframe of the most frequently used words in hashtags :
positive_df = pd.DataFrame({'Hashtags': list(positive_word_freq.keys()),'Count' : list(positive_word_freq.values())})
positive_df
```

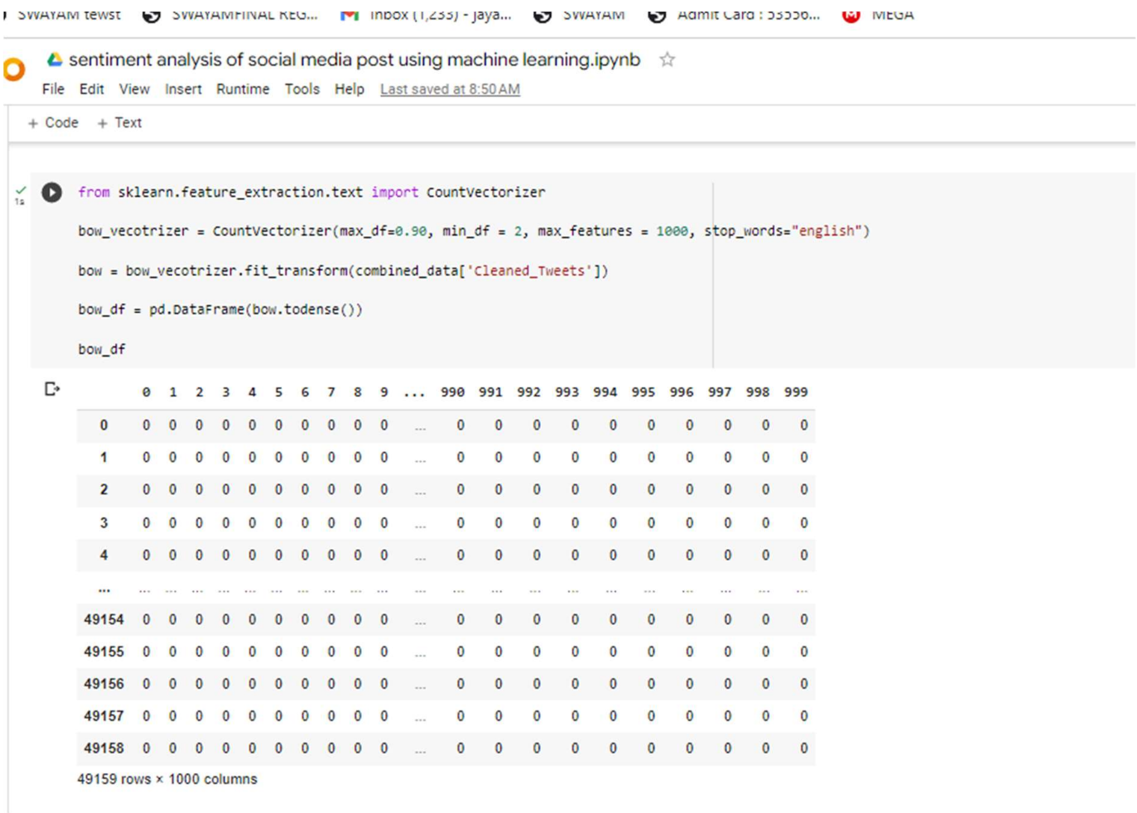
	Hashtags	Count
0	run	34
1	lyft	2
2	disapointed	1
3	getthanked	2
4	model	365
...	...	...
20744	kamp	1
20745	ucsd	1
20746	berlincitygirl	1
20747	genf	1
20748	bern	1



File Edit View Insert Runtime Tools Help Last saved at 8:50 AM
+ Code + Text

```
[27]  20748    bern    1

20749 rows × 2 columns
```

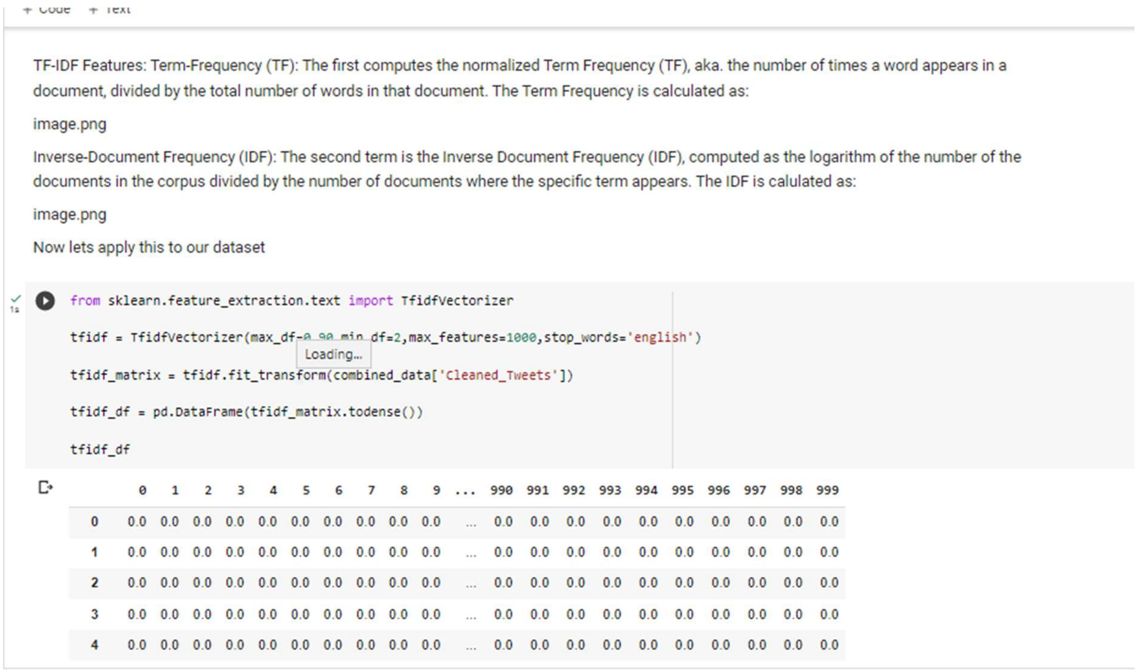
```
#Plotting the bar plot for 20 most frequent words:
positive_df_plot = positive_df.nlargest(20,columns='Count')

sns.barplot(data=positive_df_plot,y='Hashtags',x='Count')
sns.despine()
```



```python
from sklearn.feature_extraction.text import CountVectorizer

bow_vecotrizer = CountVectorizer(max_df=0.90, min_df = 2, max_features = 1000, stop_words="english")

bow = bow_vecotrizer.fit_transform(combined_data['Cleaned_Tweets'])

bow_df = pd.DataFrame(bow.todense())

bow_df
```

	0	1	2	3	4	5	6	7	8	9	...	990	991	992	993	994	995	996	997	998	999
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49154	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
49155	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
49156	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
49157	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
49158	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

49159 rows × 1000 columns



TF-IDF Features: Term-Frequency (TF): The first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document. The Term Frequency is calculated as:

image.png

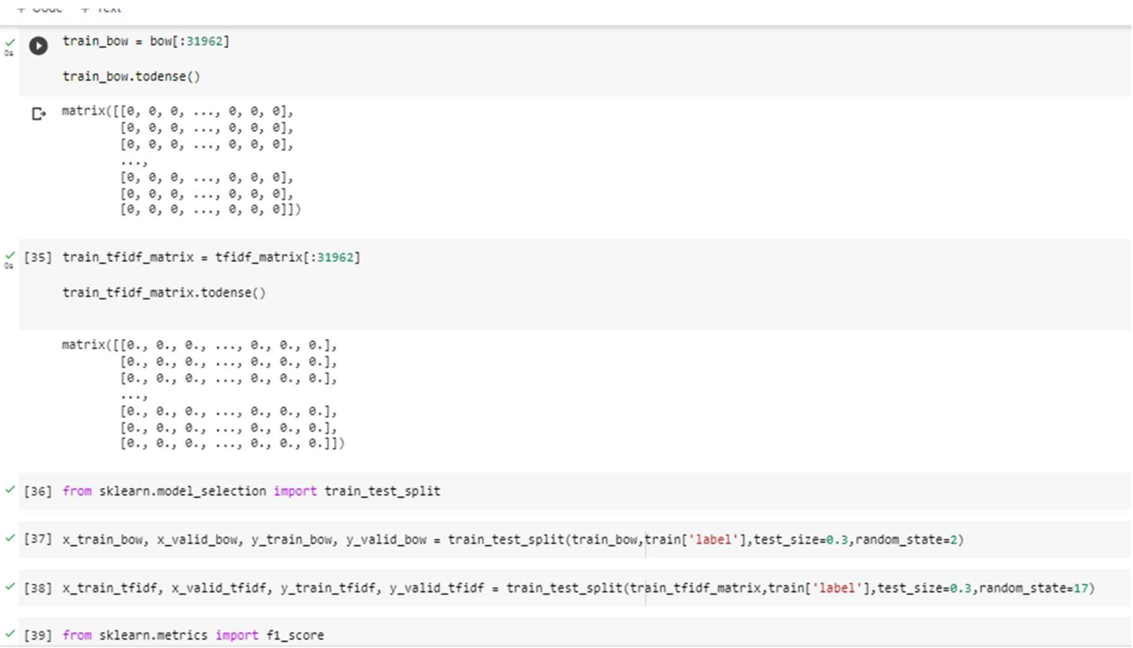
Inverse-Document Frequency (IDF): The second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears. The IDF is calulated as:

image.png

Now lets apply this to our dataset

```python
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_df=0.90,min_df=2,max_features=1000,stop_words='english')

tfidf_matrix = tfidf.fit_transform(combined_data['Cleaned_Tweets'])

tfidf_df = pd.DataFrame(tfidf_matrix.todense())

tfidf_df
```

	0	1	2	3	4	5	6	7	8	9	...	990	991	992	993	994	995	996	997	998	999
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0



```
train_bow = bow[:31962]

train_bow.todense()
```

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]])
```

```
[35] train_tfidf_matrix = tfidf_matrix[:31962]

     train_tfidf_matrix.todense()
```

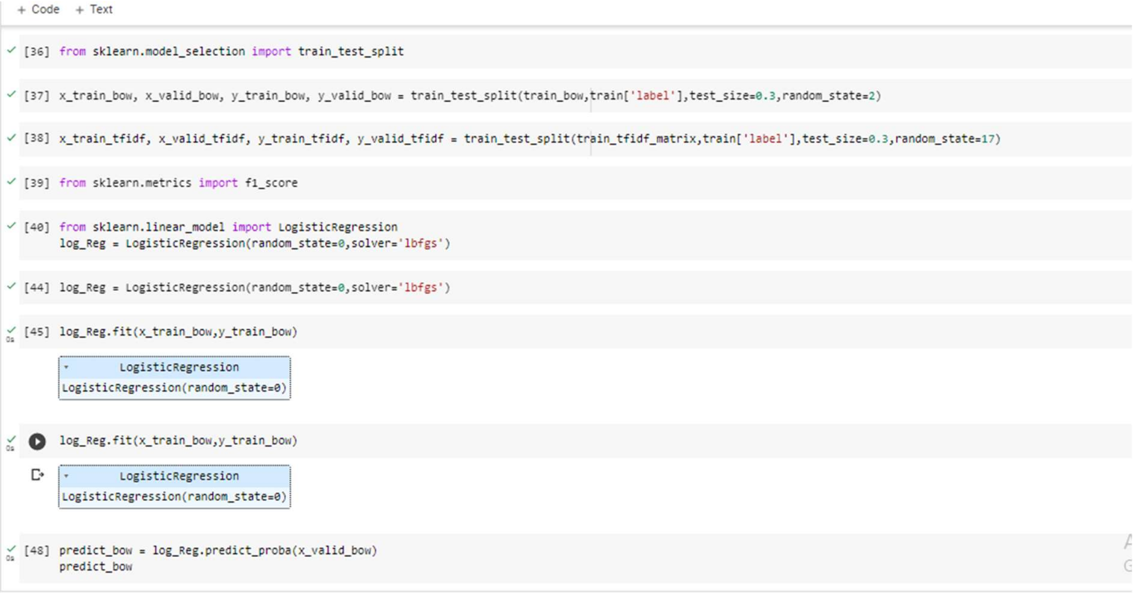
```
matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

```
[36] from sklearn.model_selection import train_test_split
```

```
[37] x_train_bow, x_valid_bow, y_train_bow, y_valid_bow = train_test_split(train_bow,train['label'],test_size=0.3,random_state=2)
```

```
[38] x_train_tfidf, x_valid_tfidf, y_train_tfidf, y_valid_tfidf = train_test_split(train_tfidf_matrix,train['label'],test_size=0.3,random_state=17)
```

```
[39] from sklearn.metrics import f1_score
```



```
[36] from sklearn.model_selection import train_test_split
```

```
[37] x_train_bow, x_valid_bow, y_train_bow, y_valid_bow = train_test_split(train_bow,train['label'],test_size=0.3,random_state=2)
```

```
[38] x_train_tfidf, x_valid_tfidf, y_train_tfidf, y_valid_tfidf = train_test_split(train_tfidf_matrix,train['label'],test_size=0.3,random_state=17)
```

```
[39] from sklearn.metrics import f1_score
```

```
[40] from sklearn.linear_model import LogisticRegression
     log_Reg = LogisticRegression(random_state=0,solver='lbfgs')
```

```
[44] log_Reg = LogisticRegression(random_state=0,solver='lbfgs')
```

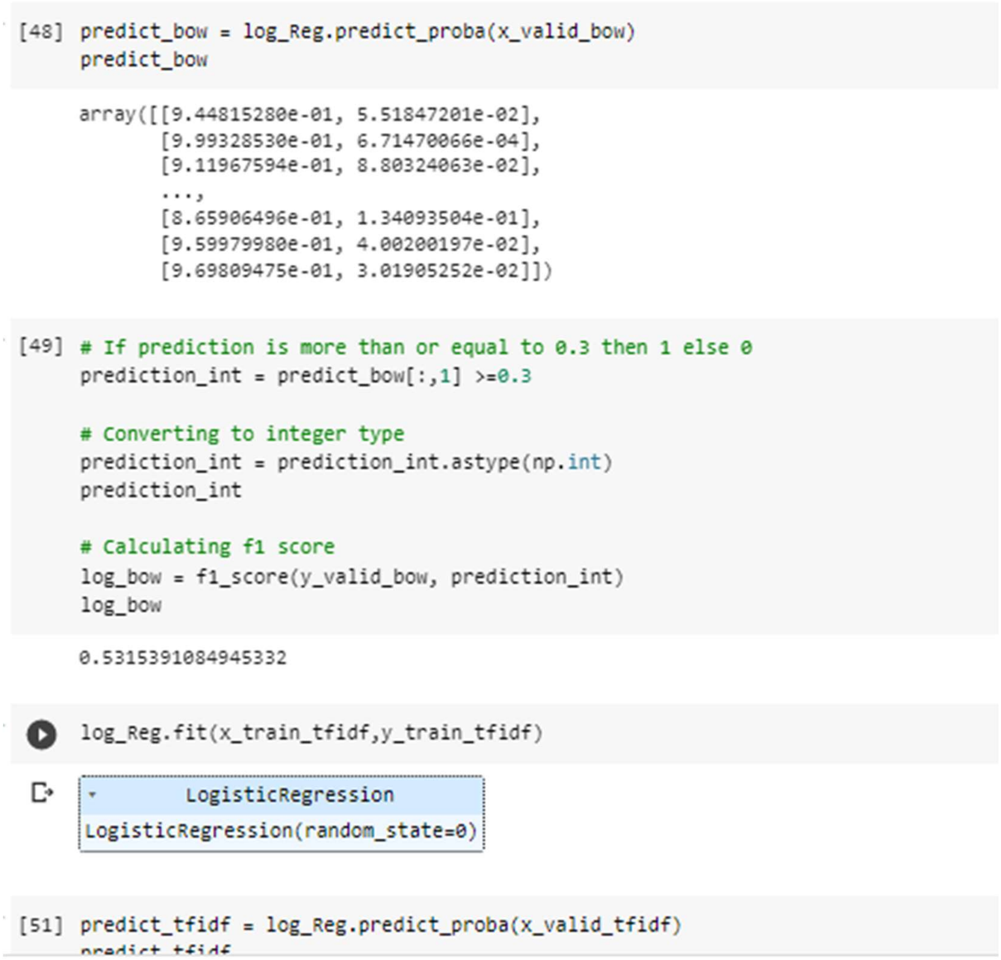
```
[45] log_Reg.fit(x_train_bow,y_train_bow)
```

```
LogisticRegression
LogisticRegression(random_state=0)
```

```
log_Reg.fit(x_train_bow,y_train_bow)
```

```
LogisticRegression
LogisticRegression(random_state=0)
```

```
[48] predict_bow = log_Reg.predict_proba(x_valid_bow)
     predict_bow
```

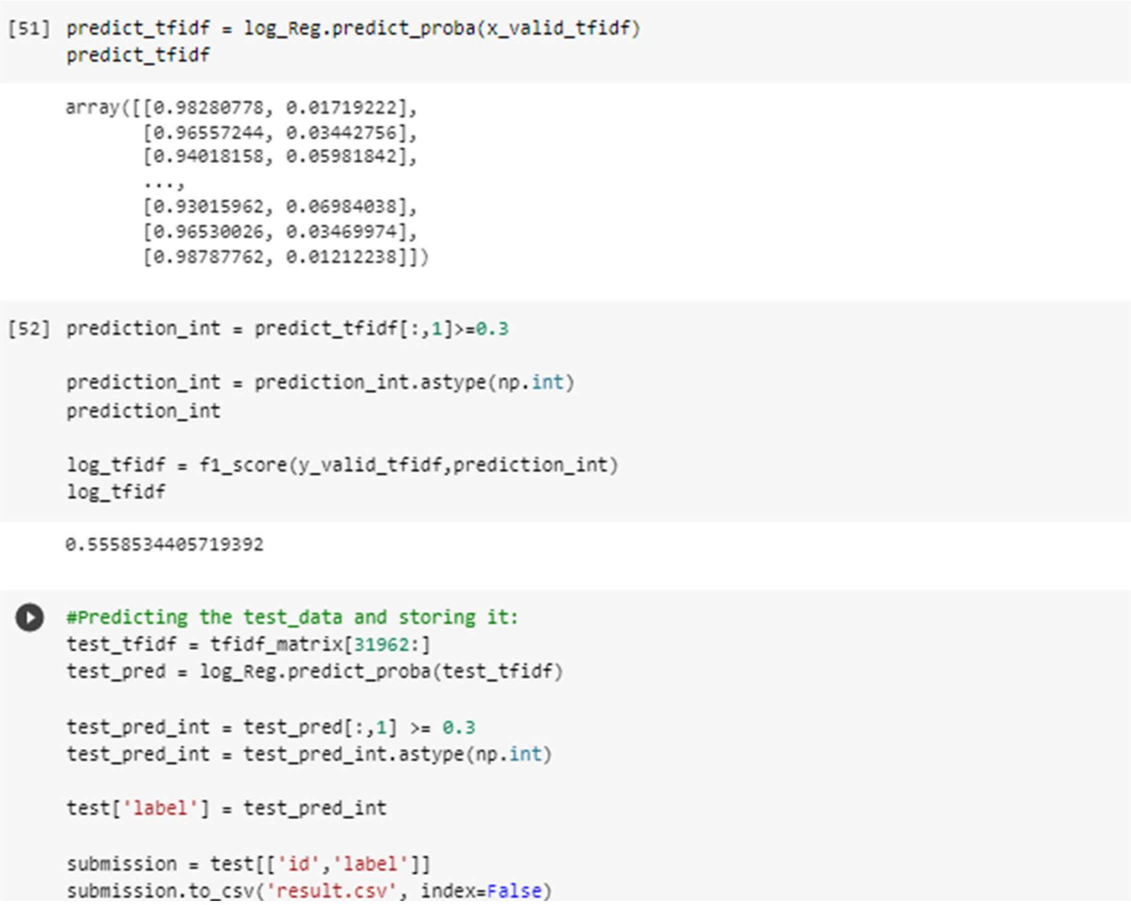


```
[48] predict_bow = log_Reg.predict_proba(x_valid_bow)
     predict_bow

     array([[9.44815280e-01, 5.51847201e-02],
            [9.99328530e-01, 6.71470066e-04],
            [9.11967594e-01, 8.80324063e-02],
            ...,
            [8.65906496e-01, 1.34093504e-01],
            [9.59979980e-01, 4.00200197e-02],
            [9.69809475e-01, 3.01905252e-02]])
```

```
[49] # If prediction is more than or equal to 0.3 then 1 else 0
     prediction_int = predict_bow[:,1] >=0.3

     # Converting to integer type
     prediction_int = prediction_int.astype(np.int)
     prediction_int

     # Calculating f1 score
     log_bow = f1_score(y_valid_bow, prediction_int)
     log_bow

     0.5315391084945332
```

```
     log_Reg.fit(x_train_tfidf,y_train_tfidf)

     LogisticRegression
     LogisticRegression(random_state=0)
```

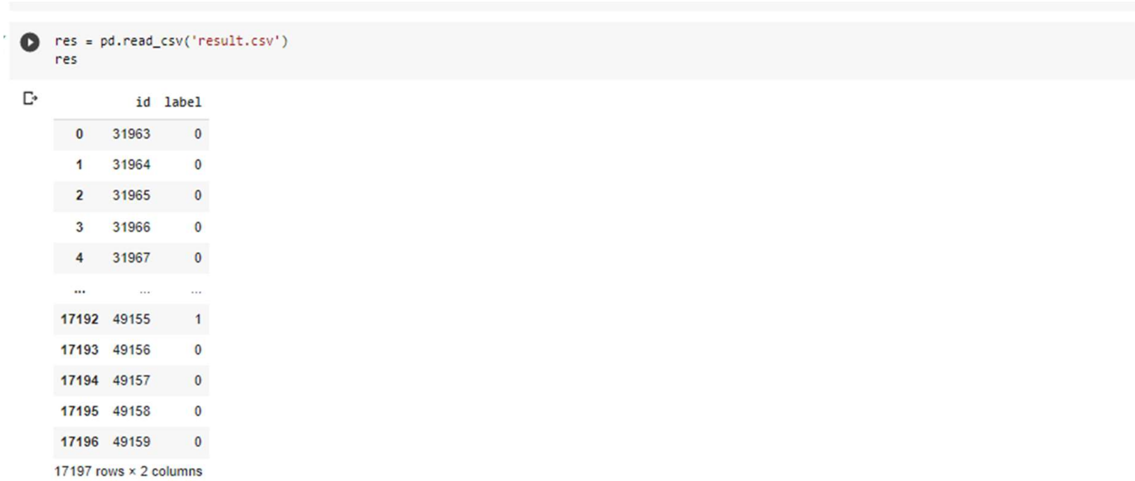
```
[51] predict_tfidf = log_Reg.predict_proba(x_valid_tfidf)
     predict_tfidf
```



```
[51] predict_tfidf = log_Reg.predict_proba(x_valid_tfidf)
     predict_tfidf

     array([[0.98280778, 0.01719222],
            [0.96557244, 0.03442756],
            [0.94018158, 0.05981842],
            ...,
            [0.93015962, 0.06984038],
            [0.96530026, 0.03469974],
            [0.98787762, 0.01212238]])
```

```
[52] prediction_int = predict_tfidf[:,1]>=0.3

     prediction_int = prediction_int.astype(np.int)
     prediction_int

     log_tfidf = f1_score(y_valid_tfidf,prediction_int)
     log_tfidf

     0.5558534405719392
```

```
#Predicting the test_data and storing it:
test_tfidf = tfidf_matrix[31962:]
test_pred = log_Reg.predict_proba(test_tfidf)

test_pred_int = test_pred[:,1] >= 0.3
test_pred_int = test_pred_int.astype(np.int)

test['label'] = test_pred_int

submission = test[['id','label']]
submission.to_csv('result.csv', index=False)
```



```
res = pd.read_csv('result.csv')
res
```

```
          id  label
0      31963      0
1      31964      0
2      31965      0
3      31966      0
4      31967      0
...      ...    ...
17192  49155      1
17193  49156      0
17194  49157      0
17195  49158      0
17196  49159      0
17197 rows × 2 columns
```

Summary: F-1 Score of Model: 0.5315391084945332 (Bag of Words) & 0.5558534405719392 (TF-IDF) using Logistic Regression